Path Planning for Autonomous Platoon Formation

O. El Ganaoui-Mourlan[1], S. Camp[2], T. Hannagan[3], V. Arora[1] , M. de Neuville[1], V. Kousournas[1],

1. IFPEN/IFP School

2: IFP School; 3 : IFP School/STELLANTIS

Abstract: This work proposes a pipeline that enables a vehicle (referred as slave throughout this paper) to autonomously join a platoon with optimized trajectory robust to uncertain traffic obstacles. A notable aspect is the use of Model Predictive Control (MPC) optimization of the path planned through a variant of Rapidly-exploring Random Trees (RRT) algorithm for the purpose of platoon formation, a combination which explores the space quickly, giving trajectories smoothened out with respect to the dynamic constraints of the vehicle, and at the same time being efficient enough for real-time implementation. The complete pipeline is simulated over various nominal and worst case scenario, qualifying the merits of the proposed methodology.

1 Introduction

Autonomous vehicles have been one of the leading driver of research in vehicle technology with promising reduction in crashes due to human-error and productivity gains for passenger (Rosenzweig, & Bartl, 2015). The traveling public can expect level 5 full automation in more than 50% of vehicles by 2030 But affordability for Level 5 autonomous cars would remain challenging in the short-term future, mainly due to the sensor costs (Kyriakidis et al, 2015). Highways represent favorably constrained environments for autonomous vehicles and in this context platooning forms a roadmap towards Automated Highway Systems (AHS), whereby a dedicated 'lead' vehicle, fully equipped with the required sensors, can guide other vehicles (minimally equipped) into a platoon and maintain it.

Platooning is a well-known research subject, with the usual goals of the study being to reduce energy-consumption in vehicles, traffic congestion (Dao et al 2013; Hobert, 2012) and to keep dynamic robustness of the platoon formed (Tuchner et al, 2017).

Autonomous co-operative platoon maneuvering as a whole has been studied using logic based rules (Lam & Katupitya, 2013). Some work has been also conducted on addition of vehicles at a merge junction on highways to an existing platoon was considered (Lu et al, 2004), but it bases its consideration on longitudinal management of vehicles. Also, both these studies lack the consideration of traffic obstacles in their work. This leads to exploration of path planning methods that explore the space around the ego that can enable platoon formation with both longitudinal and lateral maneuvers, all the while avoiding obstacles.

In the context of passenger vehicles on highways, a vehicle (referred as *slave* throughout this paper) can be enabled to autonomously join a platoon, without requiring it to have advanced perception and mapping systems. For this, a dedicated vehicle (referred as leader throughout this paper) plans and supervises through V2V, its path to integrate it into the platoon. The communication intricacies have been studied (Lam & Katupitya, 2013) and are considered as a prior. Moreover, leader-following approaches have been explored extensively previously in robots where one robot is designated as the leader and the remaining robots follow the leader's motion offset by a distance and an advantage of leader-following is that maneuvers can be specified in terms of the leader's motion (Young et al, 2001). This naturally lends to a hypothesis for slave vehicles: no advanced perception and limited computing resources. This potentially renders the vehicle desiring to join the platoon to incur only little additional cost in terms of sensors or extra devices but this is not the focus of this study.

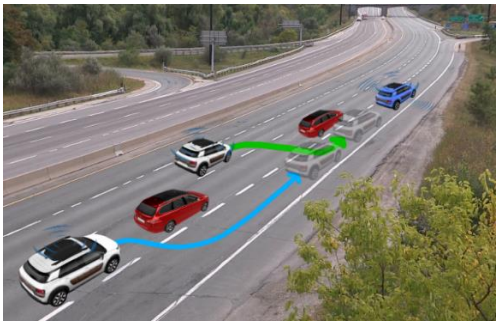

Figure 1: Platoon formation illustration

In this work, the implementation of autonomous path planning for platoon formation on highways, via a pipeline where 'slave' vehicles are guided by a 'lead' vehicle through relevant sensors and V2V communication, validated in a virtual vehicle environment, is described.

The overall goal of the implementation is to organize multiple vehicles into a platoon: find their optimum trajectories, all the while respecting the constraints: longitudinal and lateral acceleration must be below 3 m/s^2, lateral jerk must be below 5 m/s^3, all the while optimizing to minimize the time to join the platoon.

The main focus of the work is the path planning and control required for such a process, and considerations for vehicle dynamics during the manoeuvres is discussed: achieved via the combination of a variant of RRT and MPC, which does away with the conventional problems of just using RRT (Claussmann et al, 2019).

2 Platoon formation

The solution proposed in this document supposes that the vehicle participating to the platooning manoeuvre includes the architecture described next. The leading vehicle (leader) fully controls the platoon formation: it possesses all the required sensors for localization and mapping of its surroundings and the vehicles nearby. The leader doesn't need to be a L5 vehicle itself but rather is driven by a trained driver. The vehicles already integrated in the platoon or desiring to join it (slaves) have only basic sensors for safety purpose. An overall system architecture can then have the subsystems as described in Figure 2 and Appendix A.

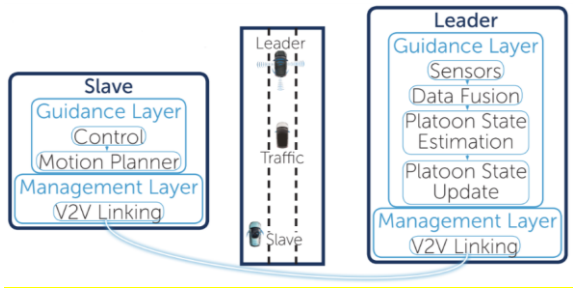

Figure 2: Vehicle system architecture for autonomous platooning

The chart in Figure 3 summarizes the flow of operations defined to build the platoon from the moment a slave indicates its intention to join it. The first step is to decide the order of the cars wanting to integrate the platoon or, in other words, their respective position in the platoon. The leader vehicle then calculates and updates the path of the slave defined as first in the ordering step. During that operation, it fully controls the slave vehicle until it reaches the desired position. This operation is then repeated for the other slaves, one slave at a time.

For each slave that has reached its position in the platoon, the leader also keeps controlling its dynamics to ensure the autonomous platoon motion.

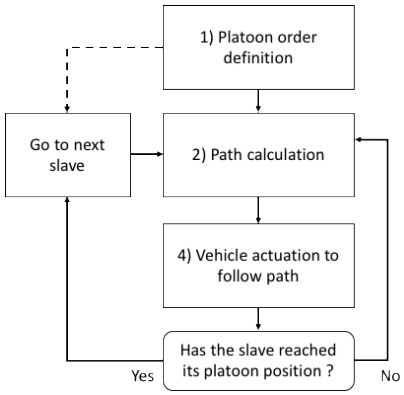

Figure 3: General operation flow of platoon formation

As Figure 2 illustrates, the major part of the platooning formation is the autonomous calculation of a path for each joining vehicle. This is done in two sequential operations:

1. Raw path calculation: this first step generates a path that complies with the configuration space: the path stays inside the road boundaries and prevent hitting any road obstacle while making its way to the target position as fast as possible.

2. Path optimization: The path calculated in (1) does not guarantee that it is physically achievable for a vehicle. The average driver won't be fond of high acceleration or jerk peaks. This leads to the second operation: path optimization. From the path defined in (1), this step will output a dynamically realistic path for the vehicle to follow.

Note that for all the following figures shown, the paths described show the vehicle's centre of gravity and not their heading. The vehicle always starts and ends up in a straight heading.

3 Path planning

3.1 Path planning with Rapid-exploring Random Trees (RRT)

Path planning finds a feasible path for the slave vehicle from its actual position to its target platoon position. In this section, a feasible path is a path that is contained into the road boundaries and that avoids other traffic obstacles including other vehicles.

There are many ways to solve this problem. Path planning has been extensively studied in graph theory and offers therefore plenty different algorithms: Dijkstra algorithm, A* algorithms, etc (LaValle, 2011).

The one chosen in this project is a variant of the Rapid- exploring Random Trees (RRT) (LaValle, 1998). This method fulfils the demands of the path planning task to be efficient (real-time capable) and safe (in face of uncertainty and sensing errors) (Frazzoli et al, 2009). The benefits of using a sampling-based method are:

- There is no need to explicitly characterize the configuration space, but instead probe the space and use collision detection on the go
- They are incremental in nature and efficient which offers the potential to real-time implementation while retaining completeness guarantees

The basic principle of RRT as would be suitable in our context of platoon formation is the following: a tree of trajectories is grown through random sampling (Cheng & LaValle, 2001) of the God's-Eye-View map created through data fusion of the sensing data. Samples are added or discarded based on the feasibility.

Although the tree expansion and hence the space exploration is very fast, the simple RRT path has a major drawback: it is highly random.

Figure 4 illustrates an RRT path calculation for the platooning problem. Even though it avoids obstacles and reaches the desired target, it is obvious that this path is not applicable in a real situation.

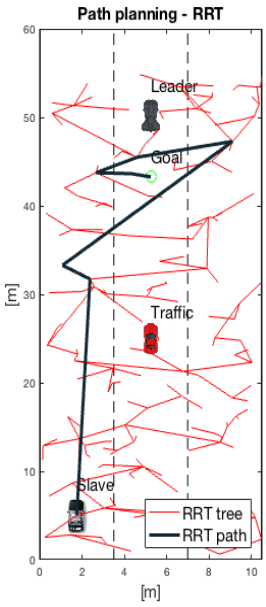

Figure 4: Path planning with RRT

3.2 Biased RRT star (RRT*)

RRT* is an evolution of the RRT algorithm (Karaman & Frazzoli, 2011). Instead of connecting each new node to its closest tree node, it connects with the node that reaches it with the least cost (distance).

To accelerate the path finding towards the target position, the sampling of the configuration space is biased: x% of the new samples are taken towards the goal position (x depends on the complexity of the situation, i.e. the number of traffic objects, the shape of the road but is usually high). This bias enables to converge faster and create straighter paths. Note that the RRT path will always be biased in the following discussion even if it is not specified.

Figure 6 illustrates biased RRT* path planning for the platooning problem. The RRT* path is less random and quickly reaches the desired position. However, this solution is not optimal. A human driver would rather pick a path on the left of the traffic vehicle.[1]

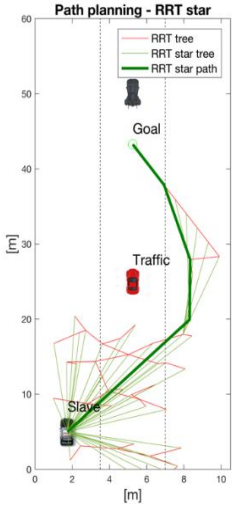

*Figure 6: Path planning with RRT**

3.3 Informed RRT* (i-RRT*)

Informed RRT* is again an evolution of the RRT* algorithm. The basic principle is to have an RRT* path search in a restricted configuration space to converge quicker to a close to optimal solution (Gammell, Srinivasa et al, 2014). The limited configuration space is taken as an ellipse whose characteristics are defined by an initial RRT* path.

Figure shows the initial RRT-star path, the ellipse derived from it and the new i-RRT* path found for the platooning problem showing desirable results.

[1] i-RRT* was implemented and tested separately. It was not added to the main pipeline at the moment of writing this document

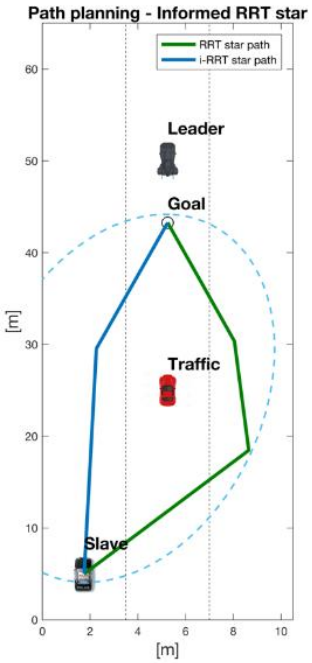

*Figure 7: Path planning with i-RRT**

4 Path Optimization using Model Predictive Control

Model Predictive Control is an optimal control technique that relies on the prediction of future outputs of the considered plant to perform an optimization under constraints of the current time-step control action. At each time-step, a cost function is minimized over a certain number of future time-steps called *prediction horizon*, calculating and applying to the prediction model a certain number of future control moves called *control horizon (*Melda Ulusoy et al*)*.

The MPC path optimization algorithm is used to generate an optimal vehicle control sequence so that the vehicle follows as accurately as possible the a priori calculated RRT path, while complying with constraints. These constraints include driver's feeling or *soft constraints* such as acceleration and jerk limits, and physical or *hard constraints* such as steering angle limits.

The core of MPC technique relies on a prediction model of the controlled system. The different vehicle prediction models and their trade-offs are discussed in Appendix B for the optimal choice of kinematic-bicycle model.

Once the RRT* algorithm has found a path it is required to determine a set of control commands to apply to the slave for it to reach the leader. The considered control variables are the steering angle δ_f and the jerk j. I-The RRT* path coordinates can be treated as setpoints for the slave lateral position. The longitudinal distance between the leader and the slave x^* is treated as a cost value to be minimized over the simulation period (prediction horizon). The control and state variables are bounded by the dynamic constraints the slave vehicle must respect. Alongside those constraints the controller aims at not being too much demanding on the control variables (jerk and steering angle) since it is a necessity for the vehicle to be gently steered and softly driven.

Now that a vehicle dynamic model has been described it is possible to use a model-based control method. Then to solve the dynamic model while complying with a certain set of constraints applied to the state and command variables an optimal control method called MPC (Model Predictive Control) is used. MPC is based on solving an optimization problem. For this study the problem is written as the minimization of the cost function J defined as follows:

$$min_{u\in U}\, J = \int_0^{t_f} q_1\left(y - y_r(x^*)\right)^2 + q_2 {x^*}^2 + q_3 j^2 + q_4 \delta_f^2 \; dt$$

$$such\ that \begin{cases} \dot{\boldsymbol{X}}(t) = \boldsymbol{A}(t)\boldsymbol{X}(t) + \boldsymbol{B}(t)\boldsymbol{U}(t) \\ \boldsymbol{Y}(t) = \boldsymbol{C}(t)\boldsymbol{X}(t) + \boldsymbol{D}(t)\boldsymbol{U}(t) \end{cases}$$

$$with\ state: \boldsymbol{X} = \begin{pmatrix} x & y & x^* & v_x & v_y & a & j & \psi & \dot{\psi} & \delta_f & x_{leader} \end{pmatrix}^t$$

$$and\ constraints \begin{cases} \mathbf{X}(t_0) = \boldsymbol{x_0} \\ t_f = 15s \\ |a| < 3\, m/s^2 \\ |j| < 5\, m/s^3 \\ v < 130\, km/h \end{cases}$$

Using:

- y the lateral vehicle coordinate [m]
- y_r the RRT* lateral coordinate [m] (Figure)
- x^* the longitudinal distance between leader and slave [m]
- x_{leader} the leader longitudinal coordinate [m]
- ψ, $\dot{\psi}$, δ_f respectively the heading angle, the yaw rate and the steering angle [rad]
- v_x, v_y respectively the slave lateral and longitudinal speed [m/s]
- $q_{i\in 1,2,3,4}$ the weights of the cost function [-]
- t_f the prediction horizon (time over which the dynamic model is solved).

The cost function can be interpreted as the cumulative sum over time of four weighed errors using the coefficients ($q_{i\in 1,2,3,4}$). Minimizing J naturally implies minimizing these cumulative errors over the prediction horizon.

The weight coefficients $q_{i\in 1,2,3,4}$ are empiracally determined by assessing the results of the

optimization solver. When setting the MPC controller the value of each q_i must be specified.

The absolute value of each coefficient taken separately has no meaning but yet their relative value with respect to each other affect the cost function behaviour during minimization. Hence to minimize an error more than the others then its associated weight in the cost function shall be increased. For instance, a large $\frac{q_1}{q_2}$ would yield a optimized path very close to the RRT* reference but would be detrimental to the time to reach the leader (since q_2 controls the distance to the leader x^*). And vice versa for a small $\frac{q_1}{q_2}$. The values listed in Table 1 have been found to yield best results.

Weight	q_1	q_2	q_3	q_4
Value	150	5	50	1

Table 1: Values of the cost function weights

The squared error terms are the following:

- $\left(y - y_r(x^*)\right)^2$: deviation between slave lateral position and the RRT* lateral coordinate.
- x^{*^2}: distance between the slave and the goal. Minimizing this state variable is the main lever to reach the leader.
- j^2: slave jerk. Minimizing the jerk prevents being too much demanding on it.
- δ_f^2 : slave steering angle. Same justification as for the jerk

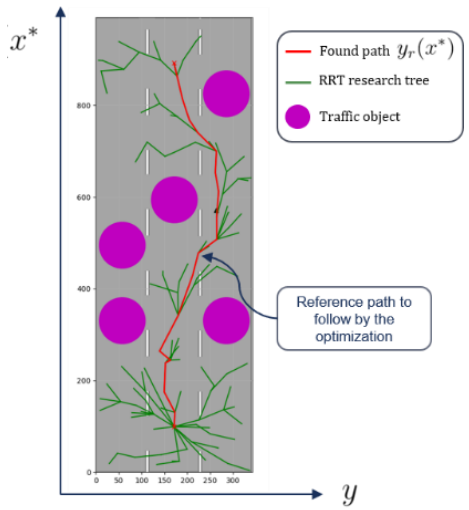

Figure 4: Raw RRT path and leader-ego x^* distance to minimize*

The dynamic equations of the system $\dot{X} = AX + BU$ can either be derived from the dynamic bicycle model or the kinematic one. This implementation of the path planner is possible in any virtual vehicle environment. The choice of the vehicle model will affect the accuracy and robustness of the path planner. Indeed, using the dynamic bicycle model will lead to a better model accuracy but will impede on computational speed. As later discussed in this paper, the higher the path planning frequency the smaller the discrepancies between actual and predicted slave position.

Next we consider solving the optimization problem practically. A python package called GEKKO, was used to solve the optimization problem which uses direct transcription method (also call "orthogonal collocation on finite elements") to minimize the cost function acting on state and control variables. These variables are being manipulated to find the optimal solution that minimizes the cost function *J*. Yet the solver plays on variables so that they respect the system dynamics. In other words, they are at any time a possible solution of the system time-based set of differential equations.

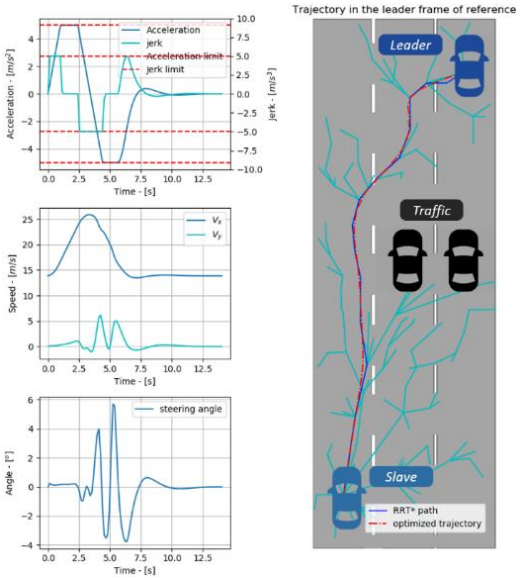

Figure 5: Results of the optimization solver using a kinematic bicycle model

The previously described optimization problem has been implemented in GEKKO (with a 150ms discretization of the kinematic bicycle model) and used the kinematic bicycle model as a first try. Results are shown on Figure 5 and applied constraints are listed in Table 2.

State variable	v_x	a	$\dot{\psi}$	j	δ_f
Minimum	0 m/s	-5m/s^2	-10°/s	-5m/s^3	-10°
Maximum	30m/s	5m/s^3	10°/s	5m/s^3	10°

Table 2: Constraints on state and control variables

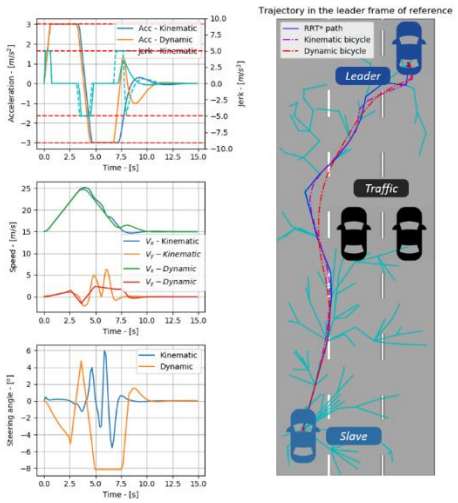

Figure 60: Comparison of the optimization solver results using kinematic and dynamic bicycle model

Figure 6 shows that for this case the constraints on acceleration, jerk and steering angle were respectively 5 m.s^(-2), 5 m.s^(-3) and 8 °. The blue dashed line represents the jerk for the dynamic bicycle model.

Beyond smoothing the jerk of raw i-RRT* path, the MPC method has provided us with the time profiles of each command variables to be adopted by the slave. The optimization solver has found a kinematically feasible solution while minimizing the deviation between the slave lateral position and i-RRT* lateral reference coordinate. The optimization has been run for a prediction horizon of 15s but a 10-second horizon has turned out to be enough to reach the leader. In the early stages of the manoeuvre the acceleration and the jerk control variables hit their limits. Indeed, since the cost function includes the distance to the leader x^*, the solver minimizes this quantity over the complete prediction horizon. This results in reaching the leader position as quickly as possible and thus the control variables are asked to operate at their limits as soon as possible.

As briefly previously discussed, the dynamic model choice will impede on path planner accuracy. Fig. 6 shows both the optimized path based on a kinematic bicycle model and the one leveraging a dynamic bicycle model. The results slightly differ in terms of command profiles, indeed the dynamic bicycle turned out to lead to a more aggressive steering angle command.

Bicycle model	Computational time
Kinematic	1.3 s
Dynamic	1.6 s

Table 3: Computational time for both models

In the final implementation the path planner would run the RRT* along with optimization algorithm several times prior to reach the leader. Throughout the manoeuvre the path planner is called numerous times at a specific frequency to run the calculation as the slave is getting closer to the leader. This is done so that the path planner could cope with a changing environment (moving traffic objects and changing leader speed). If this frequency is too low, then the time in between each optimization results is longer and the slave travels larger distances. This would lead us to face the risk of having bigger discrepancies between the model-predicted vehicle position and the actual vehicle position when beginning the next path planning calculation.

Hence to prevent the path planner from lacking accuracy one should properly set its frequency or should use a more detailed vehicle dynamics model. So it is necessary to find the best compromise between a high path planning frequency that would alleviate model inaccuracy (detrimental to computational time) and the use of a more sophisticated vehicle model that would require longer computational time as demonstrated in Table 2.

5. Simulations

After configuring the input and output signals, the platooning and optimization algorithms were implemented in a virtual vehicle environment as depicted in 11, which shows the complete pipeline.

The platoon ordering step accounts for the starting point. At the very beginning of the platooning, the RRT* path planner is run for each platooning candidate. Then, all the candidates are ranked based on their RRT* cost which is the length of the raw - RRT* path. The closest candidate will be the first to be steered and driven towards its platoon position. Once it has made its way to its goal, the platoon ordering function is called again to decide the next candidate to join the platoon. This loop is performed until all the candidates have entered the platoon.

Changing traffic, moving leader and highway curvature require to frequently repeat the path planner during a candidate attempt to reach the leader. Likewise, the imperfection of the considered dynamic model yields the same path planner execution constraint.

Figure 72 depicts a scenario involving three traffic objects and three platooning candidates. In this case the path planner frequency has been set to $f_{pp} = 5\,Hz$. The left picture is a god's-eye-view of the vehicles configuration on in a virtual vehicle environment and the right axis shows the path

planner results for different time steps. The intensity of the object's color refer to the time step of the simulation.

At each path planning step, the initial conditions used in the MPC controller are the last time step vehicle state variables. As a result, it is needed to feed the MPC controller with the updated candidate and leader states. In this example the MPC manages to correct the path infeasibilities generated by the RRT* calculation. The slave behaviour is expected to be smoother if we were to use[1] informed RRT* instead of conventional RRT*. Indeed, informed RRT* tends to correct non-relevant trajectories yielded by RRT*.

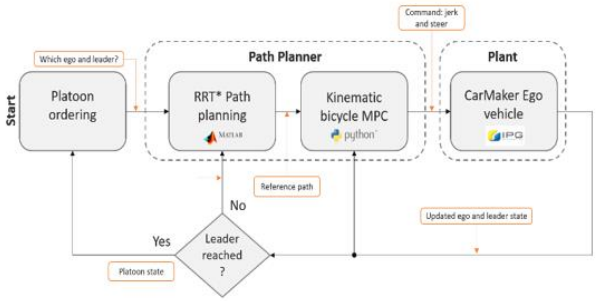

Figure 11: Path planner implementation pipeline

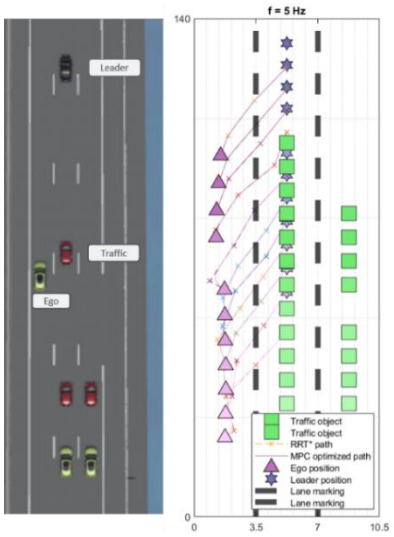

Figure 72. Configuration example with a leader, three slaves and three traffic vehicles

Figure 83 shows different frames depicting the platoon being formed. The closest slave gets called first. It finds a trajectory around the traffic in front of it and joins the leader. Then the slave from the back is called second. It averts the two traffic vehicles in front of it as well as the third vehicle and aims to drive behind the first slave. Finally, the third slave is allowed to join the platoon and finishes the manoeuvre. Each slave had to abide by the constraints given to it as well as keep a safe distance from both the traffic vehicles as well as the other slaves in the platoon.

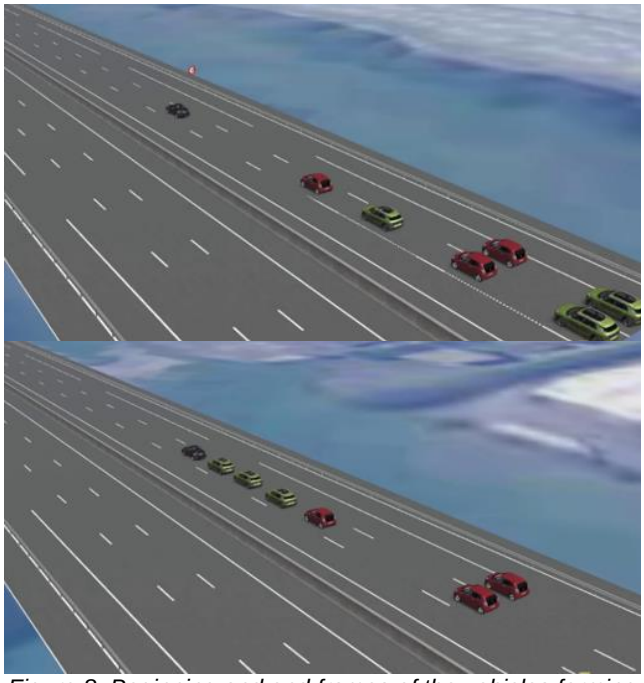

Figure 8: Beginning and end frames of the vehicles forming the platoon

6. Discussion and Conclusion

This paper has described an implementation of a platoon formation pipeline on highways lending itself towards the efforts for automated highway systems, with varying dynamic initial positions and accounting for obstacles. Initial results simulated over in a virtual vehicle environment show potential of the method. In particular, the efficacy of use of a variant of RRT for the trajectories along with MPC is promising. Several shortcomings still need to be treated: the pipeline needs to be more robust and several scenarios are still to be studied, such as the leader being behind slave vehicles, or all three lanes being completely blocked by traffic. The path planner considered no highway driving protocols, which should be implemented in the future, such as knowing the legal way to cross lanes. Finally, although the RRT algorithm provides us with an obstacle-free trajectory, the MPC controller yields a very close but different still trajectory. There is thus a risk of colliding with obstacles if the path planner frequency is too low.

References

Driving Juan Rosenzweig, Michael Bartl (2015). *A Review and Analysis of Literature on Autonomous Driving* [https://michaelbartl.com/wp-content/uploads/Lit-Review-AD_MoI.pdf]

Kyriakidis, M.; Happee, R.; De Winter, J.C.F. (2015) *Public opinion on automated driving: Results of an international questionnaire among 5000 respondents*. Transp. Res. Part F Traffic Psychol. Behav. 2015, 32, 127–140.

Thanh-Son Dao, Jan Paul Huissoon, Christopher Michael Clark (2013). *A strategy for optimization of cooperative platoon formation.* Int. J. Vehicle Information and Communication Systems, 2013, Vol. x, No. x, xxxx [10.1504/IJVICS.2013.055766]

Hobert, L.H.X. (2012) *A study on platoon formations and reliable communication in vehicle platoons*.

Tuchner, A.; Haddad, J. (2017). *Vehicle platoon formation using interpolating control: A laboratory experimental analysis.* Transp. Res. Part C Emerg. Technol. 2017, 84, 21–47.

Stanley Lam, Jayantha Katupitiya (2013). *Cooperative Autonomous Platoon Maneuvers on Highways.* IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) Wollongong, Australia, July 9-12, 2013

Xiao-Yun Lu , Han-Shue Tan , Steven E. Shladover & J. Karl Hedrick (2004). *Automated Vehicle Merging Maneuver Implementation for AHS*, Vehicle System Dynamics, 41:2, 85-107, DOI: 10.1076/vesd.41.2.85.26497 [https://doi.org/10.1076/vesd.41.2.85.26497]

B. J. Young, R. W. Beard and J. M. Kelsey (2001). *A control scheme for improving multi-vehicle formation maneuvers*. Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148), Arlington, VA, USA, 2001, pp. 704-709 vol.2, doi: 10.1109/ACC.2001.945797.

Claussmann, L., Revilloud, M., Gruyer, D., & Glaser, S. (2019). A Review of Motion Planning for Highway Autonomous Driving. IEEE Transactions on Intelligent Transportation Systems, 1–23. doi:10.1109/tits.2019.2913998

LaValle, S. M. (2011). *Motion Planning: The Essentials*, 18(1):79:89, from IEEE Robotics and Automation Magazine from https://ieeexplore.ieee.org/document/5751929

Lavalle, S. (1998). *Rapidly-exploring random trees: a new tool for path planning*, from the annual research report. http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=A1AB62F78BC767D0F82777B22AA4B79C?doi=10.1.1.35.1853

Emilio Frazzoli , Yoshiaki Kuwata, Gaston Fiore Real-time (2009). *Motion Planning with Applications to Autonomous Urban Driving.* IEEE Transactions on Control Systems Technology, Vol. 17, No. 5

Cheng, P., & LaValle, S. M. (2001). *Reducing metric sensitivity in randomized trajectory design*. Retrieved 6 20, 2019, from https://pdfs.semanticscholar.org/15e7/f233088f52dffadc0b07b5ff90ad69a71323.pdf

Karaman, S., Frazzoli, E. (2011). *Sampling-based algorithms for Optimal Motions Planning,* International Journal of Robotics Research. https://arxiv.org/abs/1105.1186

Gammell, J., Srinivasa, S., et al (2014). *Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic,* 2997-3004, International Conference on Intelligent Robots and Systems (IROS 2014). https://arxiv.org/abs/1404.2334

Committee, I. T. (n.d.). *European Agreement on Main International Traffic Arteries (AGR)*. Retrieved 6 15, 2019, from United Nations Economic Commission for Europe: http://live.unece.org/fileadmin/DAM/trans/conventn/ECE-TRANS-SC1-384e.pdf

Kong, J., Pfeiffer, M., Schildbach, G., & Borrelli, F. (2015). *Kinematic and dynamic vehicle models for autonomous driving control design*. Retrieved 6 20, 2019, from http://me.berkeley.edu/~frborrel/pdfpub/iv_kinematicmpc_jason.pdf

Melda Ulusoy, M. (n.d.). *Understanding Model Predictive Control, Part 3: MPC Design Parameters Video - MATLAB & Simulink*. Retrieved from https://fr.mathworks.com/videos/understanding-model-predictive-control-part-3-mpc-design-parameters-1530607670393.html

Steven Waslander, J. K. (n.d.). *Lesson 5: Lateral Dynamics of Bicycle Model - Module 4: Vehicle Dynamic Modeling | Coursera*. Retrieved from https://www.coursera.org/lecture/intro-self-driving-cars/lesson-5-lateral-dynamics-of-bicycle-model-1Opvo

Sussex, U. o. (n.d.). *Microsoft PowerPoint - MAS_03a_LateralVehicleDynamics.ppt*. Retrieved from http://users.sussex.ac.uk/~tafb8/mas/MAS_03a_LateralVehicleDynamics.pdf

APPENDIX A

Leader:

- A **Sensors** subsystem collects and translates all the data sensed through various sensors such as lidars, cameras, radars, etc.
- A **Data Fusion** block is responsible for using the data from sensors to build a 'God's-Eye-View'
- A **Platoon State Estimation** block locates the nearby slaves, traffic vehicles and decides the order of the slaves to join the platoon, selecting the respective slave for motion planning
- A **Platoon State Update** module produces a dynamically feasible trajectory for the selected slave to move from its current position to the required position in platoon in form of way-points (position, velocity, heading) w.r.t the slave's frame of reference.
- A **V2V Linking** block is responsible for sending the planned trajectory and receiving the current state of the slaves

Slave:

- The **V2V Linking** block is responsible for receiving the planned trajectory and sending the current state of the slaves
- The **Motion Planner** block accounts for safety and can over-ride the trajectory waypoints if necessary
- The **Controller** executes motion control to track the desired path and velocity profiles from the Leader's Platoon State Update module

APPENDIX B

Prediction model

The choice of the prediction model is a trade-off between accuracy and computational effort. However, an accurate prediction of the system future behavior allows to enlarge the time-step, thus reducing the computational effort. The benefits of using a more accurate prediction model will be discussed in the next part.

4.1.1.1 Dot model:

The simplest model of a car is the *dot model*. It consists in describing the motion of a massless dot:

$$\begin{cases} v_x = \dot{x} \\ a_x = \dot{v_x} \\ j_x = \dot{a_x} \end{cases} and \begin{cases} v_y = \dot{y} \\ a_y = \dot{v_y} \\ j_y = \dot{a_y} \end{cases} \quad [1]$$

It can also be written under the state space form:

$$\boldsymbol{X} = (x \quad v_x \quad a_x \quad y \quad v_y \quad a_y)^t \; and \; \boldsymbol{u} = (j_x \quad j_y)^t$$

$$\dot{\boldsymbol{X}} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A} \end{pmatrix} \boldsymbol{X} + \begin{pmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B} \end{pmatrix} \boldsymbol{u} \; [2]$$

$$with \; \boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \; and \; \boldsymbol{B} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

This model does not accurately represent the kinematics of a car nor its dynamics but allows to introduce constraints and to compute an optimal control sequence of the longitudinal and lateral jerks over the path.

Kinematic bicycle model

A more accurate model is the *kinematic bicycle model* (or Ackermann model). This model simplifies a car's geometry lumping its two front and rear wheels into a single front and rear wheel respectively. It does not account for mass nor height of the vehicle. Figure 9 shows a schematic view of the bicycle model. The frame of reference (x_b, y_b) is fixed on vehicle center of gravity Cg and x_b always points in the vehicle's heading direction. The frame of reference (x_i, y_i) is the inertial absolute frame. The angle $(\widehat{x_i, x_b})$ is the yaw angle ψ. The steering angle at the front wheels is δ, and the angle $(\widehat{x_b, V})$ is the slip angle β. l_f, l_r are respectively the lengths between the front and rear axle and the center of gravity.

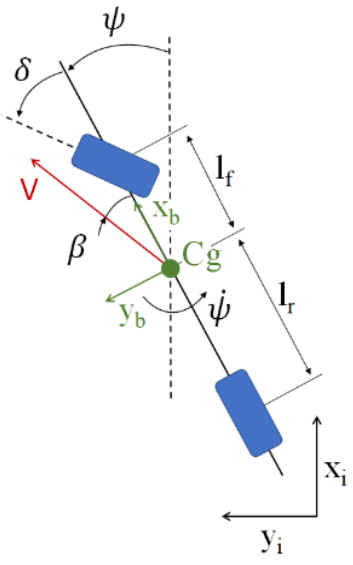

Figure 9: Bicycle model

This model is governed by the following set of equations (written at the center of gravity shown in Figure 9:

$$\begin{cases} \dot{x_i} = V \times cos(\psi + \beta) \\ \dot{y_i} = V \times sin(\psi + \beta) \\ \dot{\psi} = \frac{V}{l_r} \times sin(\beta) \\ \dot{V} = a \\ \dot{a} = j \\ \beta = tan^{-1}\left(\frac{l_r}{l_f + l_r} \times tan(\delta)\right) \end{cases} \quad [3]$$

$$Inputs: \ \boldsymbol{u} = (j \quad \delta)^t$$

This model represents more accurately the kinematics of a car, accounting for its dimensions and the geometry of wheels steering. Its control variables, the jerk and the steering angle at the wheels are closer from reality.

Decoupled dynamic bicycle model

The last prediction model is a *decoupled dynamic bicycle model*. For simplification purposes, it has been chosen to decouple the longitudinal and lateral dynamics of the vehicle into two separate models that can interact. A dynamic model is the most accurate representation of a car since it is based on a *kinematic bicycle model* to which it adds mass and tires consideration.

Lateral dynamic bicycle model:

The tires are considered as an object that generates a lateral force as a function of the slip angle between their heading angle and the velocity angle. Experimental data shows that for small slip angles (typically less than 8°), the relation is linear. (Steven Waslander, n.d.)

On Figure 10, the slope of the force versus the slip angle at zero is the *cornering stiffness* of the tire C_x, expressed in $N.rad^{-1}$.

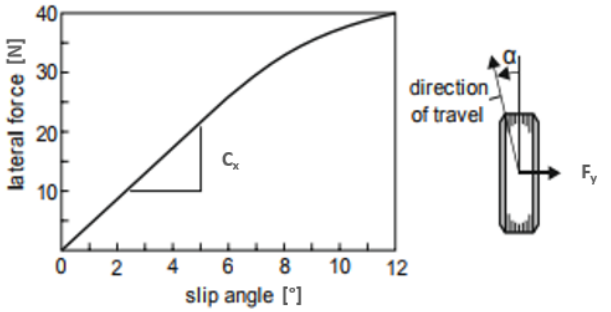

Figure 10: Relation between tire slip angle and lateral force (Steven Waslander, n.d.)

This model will be used for normal highway driving conditions. Therefore, it is assumed that the slip angle β between the velocity angle and the heading angle of the vehicle shown on Figure 9 remains small (under 5° or 0.1 rad). This assumption allows to write the two following approximations:

$$\begin{cases} V = \frac{\dot{x_b}}{cos(\beta)} \approx \dot{x_b} \\ \dot{y_b} = V\, sin(\beta) \approx \dot{x_b} \times \beta \end{cases} \quad [4]$$

The governing system of equations is the following (Sussex, s.d.):

$$\begin{cases} \dot{y_i} = \dot{y_b}\, cos(\psi) + \dot{x_b}\, sin(\psi) \\ \ddot{y_b} = -\frac{2(C_r + C_f)}{m\, \dot{x_b}} \dot{y_b} - \left(\frac{2(C_f l_f - C_r l_r)}{m\, \dot{x_b}} + \dot{x_b}\right)\dot{\psi} + \frac{2C_f}{m}\delta \\ \ddot{\psi} = -\frac{2(C_f l_f - C_r l_r)}{I_z\, \dot{x_b}} \dot{y_b} - \frac{2(C_f l_f^2 + C_r l_r^2)}{I_z\, \dot{x_b}} \dot{\psi} + \frac{2C_f l_f}{I_z}\delta \end{cases} \quad [5]$$

$$States: \boldsymbol{X} = (y_i \quad \dot{y_b} \quad \psi \quad \dot{\psi})^t$$

$$Input: \boldsymbol{u} = \delta$$

Using:

- C_r, C_f the cornering stiffness of respectively the rear and front tire
- I_z the moment of inertia around z axis

For highway lane keeping, this system of equations can be linearized using the assumption that the yaw angle ψ remains small (under 5° or 0.1 rad). The lateral position in the inertial frame becomes:

$$\dot{y_i} \approx \dot{y_b} + \dot{x_b}\psi$$

The model can then be written under the following state-space form:

$$\dot{\boldsymbol{X}} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{B}\boldsymbol{u}$$

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & \dot{x_b} & 0 \\ 0 & -\frac{2(C_r + C_f)}{m\, \dot{x_b}} & 0 & -\frac{2(C_f l_f - C_r l_r)}{m\, \dot{x_b}} - \dot{x_b} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{2(C_f l_f - C_r l_r)}{I_z\, \dot{x_b}} & 0 & -\frac{2(C_f l_f^2 + C_r l_r^2)}{I_z\, \dot{x_b}} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 \\ \frac{2C_f}{m} \\ 0 \\ \frac{2(C_f l_f)}{I_z} \end{bmatrix} \quad [6]$$

Remarks:

- The yaw angle is an important parameter of the model, it should therefore be among the outputs of the model. In practice, it can be measured through accelerometer, gyroscope and compass sensors fusion.
- This model depends on the longitudinal speed $\dot{x_b}$. It needs to be updated at each time step with the actual speed calculated by the longitudinal dynamics model.

Longitudinal dynamics model

The longitudinal dynamics model implements the basic principle of dynamics, accounting for three forces: the traction force at the wheels, which is assumed to be always in the heading direction of the vehicle (valid even for a front wheel drive vehicle on a highway), the aerodynamic forces and the rolling resistance forces.

$$F_{traction} - F_{aero} - F_{rr} = m \times \ddot{x_b} \qquad [7]$$

$$\ddot{x_b} = \frac{T}{m \times R_{wheel}} - \frac{1}{2 \times m} \times \rho \times SCx \times \dot{x_b}^2 - \frac{C_{rr}}{1000}$$

$$input: Torque\ at\ the\ wheels\ T$$

Using:

- m the vehicle mass [kg]
- R_{wheel} the wheel radius [m]
- ρ the air density [kg/m^3]
- SCx the vehicle surface times its aerodynamic drag coefficient [m^2]
- C_{rr} the rolling resistance coefficient [N/t]

This model computes the longitudinal velocity and feeds it to the lateral dynamics model. It also allows to optimize the torque demand profile along the path.

The model can be linearized using either:

- An aerodynamic drag map
- A linearization of the aerodynamic forces around an operating point $\dot{x_{b_0}}$:

$$\mathrm{F_{aero}} \approx \frac{1}{2} \times \rho \times SCx \times \dot{x_{b_0}}^2 + \left(\dot{x_b} - \dot{x_{b_0}}\right) \times \rho \times SCx$$

In our case the longitudinal dynamic model is not conventionally used to solve vehicle speed given a set of forces that act on it. In the next section, the MPC control method is further explained, which enables to determine the speed profile rather than using the above model. Consequently, this model will provide a mean to solve for $\boldsymbol{F_{traction}}$ based on the speed profile found with the control method. Hence giving a way to specify the required engine torque based on the acceleration profile found by the MPC.